Data Mining (APC308)
Lab: Introduction to Classification

April 2022

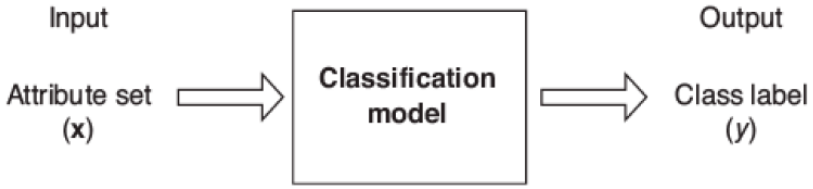
Learning outcomes

At the end of this lab session, you will be able to:

1. Describe the basic concept of classification.
2. Use relevant tools to build a decision tree classifier.

1 Basic Concepts

- Recall that at the onset of this module, we discussed some use cases of data mining such as **classification**, clustering, and association rule mining among many others. In the next few sessions, we will focus on some classification techniques/algorithms. Many of the algorithms are commonly used in areas such as data mining, machine learning, data science, etc. Therefore, the benefit of acquainting yourself with them goes a long way.
- Let me also remind you of our discussion on data mining categories such as **predictive** data mining and **descriptive** data mining.
- Classification is one of the data mining techniques that assigns objects to *predefined* categories (classes). Consider the detection of spam emails for instance. Based on the contents of the email messages and header, you categorize each email message as *spam* or *not spam*. Another example is classifying bank customers who are applying for loan as *safe* or *risky*. You could think of a number of application areas of classification. In the process, you are basically deciding what class/category each instance falls in based on some input attributes (features). The following diagram depicts this:



- The task of mapping input attribute set X to one of the predefined class labels Y is the essence of classification problems. A **model** or **classifier** is constructed to predict class label (categorical) from the input data. In case you are wondering about prediction of continuous numeric values (e.g. how much will a house rent be? How much a customer will spend?), have

patience. We will come to this in another session. If you want to quench your curiosity (did it really kill the cat by the way? :)), *regression analysis* is one of the prediction methods you can explore at your own pace.

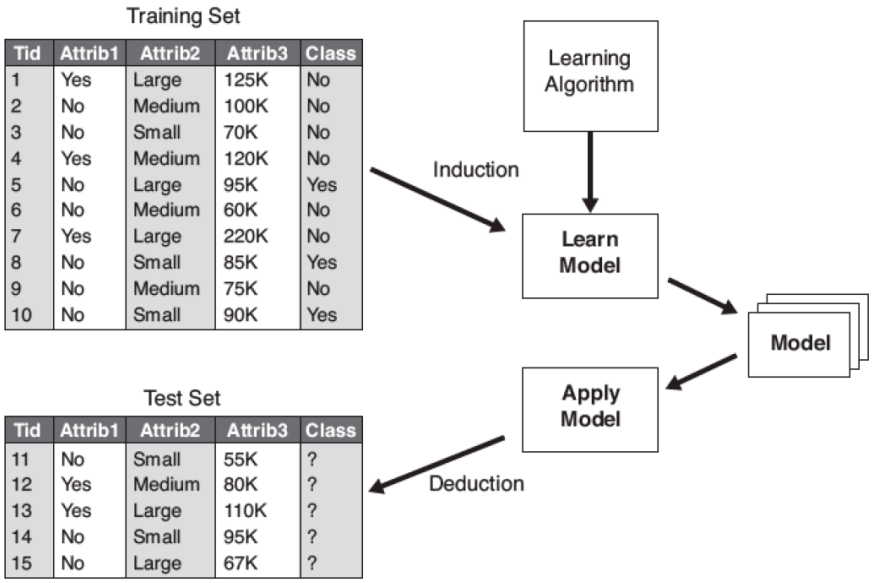
- Classification is one form of **supervised** learning technique. In such learning, the learning of the classifier is guided (supervised) by a predefined class label each instance belongs to. On the other hand, in another learning type named **unsupervised** learning, the class label of each tuple or the number of classes they must belong to is NOT known or given. The learner algorithm figures this out from the input features. Clustering is one such technique we shall discuss later on.
- A `classifier` is simply a methodic approach to building classification models from input features of a dataset. There are a number of classification algorithms commonly used in data mining/ML problems. Some examples include decision tree classifier, rule-based classifier, Support Vector Machines (SVM), naïve Bayes classifier, and neural networks. We shall get started with the first one in this session.

General Approach

Generally, classification can be seen as a two-step process:

1. **Learning (training) phase** : a classification model is built from a dataset by using a classifier. Recall that in the last lab session, we made a distinction between **training set** and **test set**. A training set is used to "teach" or train a classification algorithm. The other set is used to test how well the classifier performs on an untrained (previously-unseen) dataset, hence the name. There is another set named *validation set* which we shall not discuss today.
2. **Classification step** : the trained model is now used to predict the class label of a dataset containing unknown class labels.

- This general approach is shown in the figure below (image source: textbook).



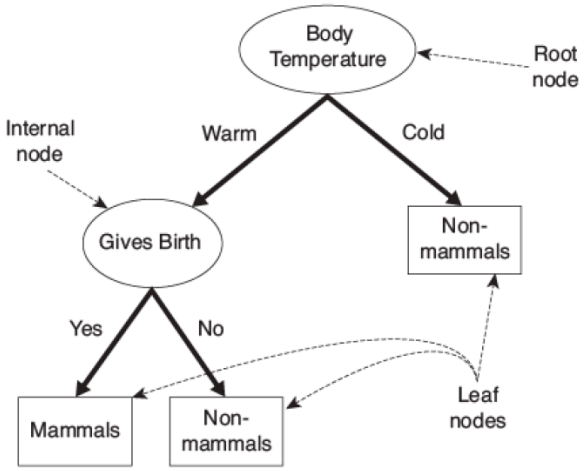
- The performance of a model can be evaluated using parameters such as **accuracy** and **error rate**. Accuracy refers to the percentage of test set instances that have correctly been classified into their categories (i.e. ratio of correct predictions and total number of predictions). The more the accuracy the better. `Error rate` is the ratio of wrong predictions and total number of predictions.

Decision Tree Classifier

- It is one of the classification techniques we can use to build our model.
- It is a flow-chart like tree structure consisting of nodes and directed edges. It consists of the following nodes:

1. **Root node**: the top most node with no incoming edges and zero or more outgoing edges.
2. **Internal (non-leaf) nodes**: indicates a test (check) on an attribute. It has exactly one incoming edge and two or more outgoing edges.
3. **Leaf node**: indicates a class (target) label. It has exactly one incoming edge and no outgoing edges.

- An example decision tree is shown below.



- A number of algorithms have been developed to build a decision tree from a set of attributes. Some include: Hunt's algorithm, ID3 (Iterative Dichotomiser), C4.5 (successor of ID3), and CART (Classification and Regression Trees). Many of the algorithms construct a decision tree in a top-down recursive, divide-and-conquer manner. Discussion of the algorithms is not the scope of this session.
- There are measures (a.k.a. attribute selection measures) that can be used to determine the attribute that results in the best split. The attribute having the best score in a chosen measurement is chosen as the splitting attribute. Some popular attribute selection measures include *information gain (entropy)*, *gain ratio*, and *Gini index*. Decision tree algorithms may differ in the attribute selection measure they use. For instance, ID3 uses information gain, C4.5 uses gain ratio, and CART uses Gini index.

2 Getting Started

2.1 Using Orange

a. Basic workflow of Tree

b. Rank demo (attribute selection)

c. Data Sampler demo

d. Evaluation widgets

Activity!

Use a different dataset included in Orange (such as titanic, heart_disease, zoo) and build a decision tree model. You may have to do preprocessing depending on the dataset. Analyze its performance.

2.2 Using Coding

A typical workflow of model building involves the following:

1. Loading the dataset and doing EDA.
2. Data preprocessing (preparation)
3. Model building using a training set
4. Model testing and evaluation

Let us see this using the iris dataset you are already familiar with.

[ ]:
```
import pandas as pd
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt
import missingno as mn
```

1. Data Loading

[ ]:
```
data = pd.read_csv('iris.csv')

data
```

Exploratory Data Analysis

[ ]:
```
data.info()
```

[ ]:
```
data.head(10)
```

[ ]:
```
# visual inspection
mn.matrix(data)
```

2. Data Preprocessing

- If there is any data quality issue (such as missing values, data type issue), deal with each first. Always remember the GIGO principle.
- Segregate the input attributes (features) from the class label (target attribute). In the above case, the first 4 attributes are the features that determine the category of a particular iris species.
- We then split the dataset into training set and test set. Recall that you have done this in the previous lab session.

[ ]:
```
# segregation of features from target attribute

input_cols = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']

x = data[input_cols]        # features. An alternative --> x = data.iloc[:, 0:4]

y = data['class']           # target / class attribute
```

[ ]:
```
# split the dataset into two subsets, say 80:20.
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.
 ↪train_test_split.html

x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8)

x_train.shape
y_train.shape
x_test.shape
y_test.shape
```

3. Model Building (Fitting)

- Sklearn provides a number of classifiers including decision tree.
- It is highly customizable. See its documentation page at the official site.
- The basic workflow now is to **fit** our model with the training set and let it **predict** a class label of our test set.

[ ]:
```
treeModel = tree.DecisionTreeClassifier()            # instantiate

treeModel.fit(x_train, y_train)

predicted = treeModel.predict(x_test)                  # why not y_test?

predicted
```

4. Model Evaluation

Congratulations! You have built your first classifier and predicted a target class from features. You can notice how Sklearn simplifies the daunting task into few lines of code. But how is its performance? We can check its accuracy and proportion of correct and wrong classifications made.

- Sklearn has a provision for many metrics. See this for more info. Let us just look at one of them named **accuracy_score()**.

[ ]:
```
metrics.accuracy_score(y_test, predicted)
```

[ ]:
```
# Let's now plot of a confusion matrix
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.
 ↪confusion_matrix.html

metrics.plot_confusion_matrix(treeModel, x_test, y_test)  # interpret the output
```

Visualizing a Tree

- We can visualize a decision tree if need be.
- We can use Sklearn's **plot_tree()** or use other external libraries such as GraphViz and Dtreeviz.
- Let us see plot_tree() in action on a default iris dataset Sklearn provides.

[ ]:
```
from sklearn.datasets import load_iris

fig, ax = plt.subplots(figsize=(20, 20))                    # plotting area

iris = load_iris()

X, Y = iris.data, iris.target                               # no splitting here

treeModel2 = tree.DecisionTreeClassifier()

treeModel2 = treeModel2.fit(X, Y)

tree.plot_tree(treeModel2)
```

Activity!

- Use a different dataset this time and practice building a tree classifier.
- You may explore other tree visualization libraries as well.